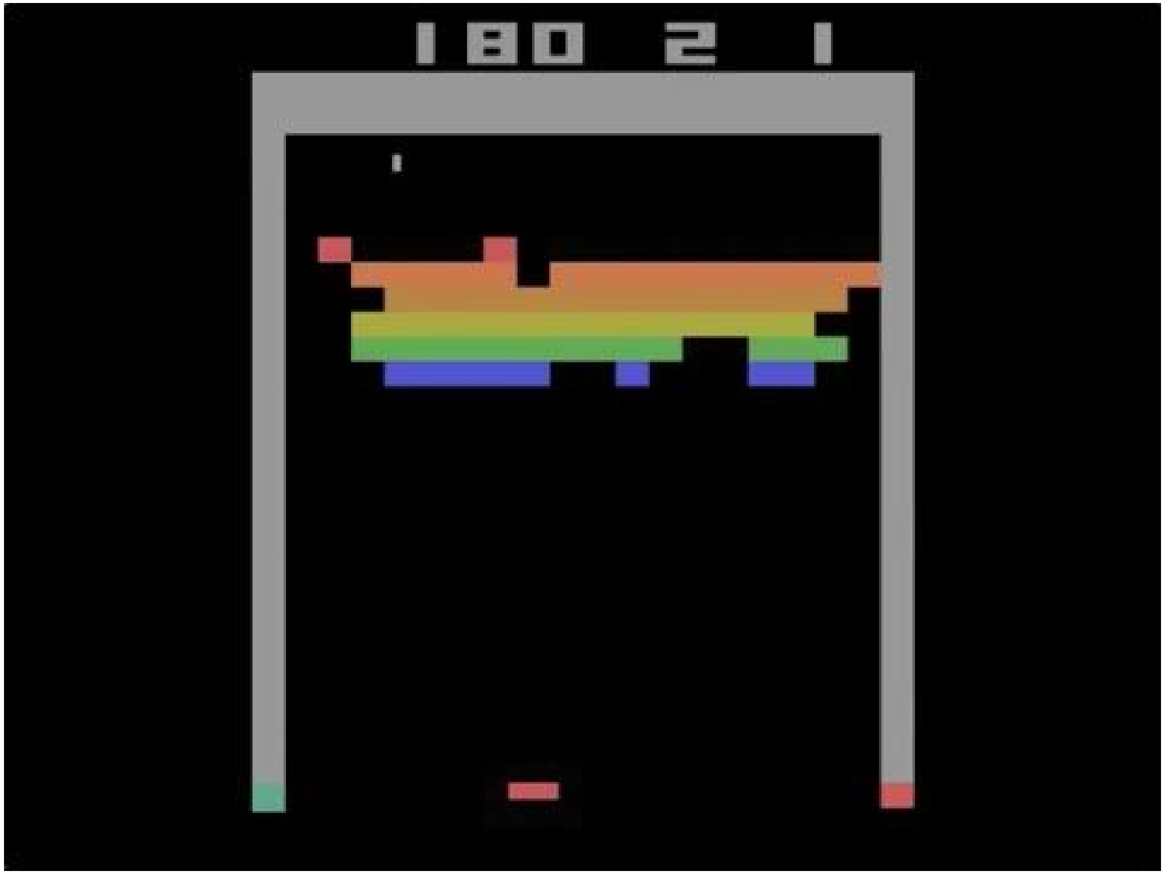
A Practical Introduction to Deep Reinforcement Learning

Chris Lu
Hosted by Machine Learning @ Berkeley

What is Reinforcement Learning?

Video Example: DQN

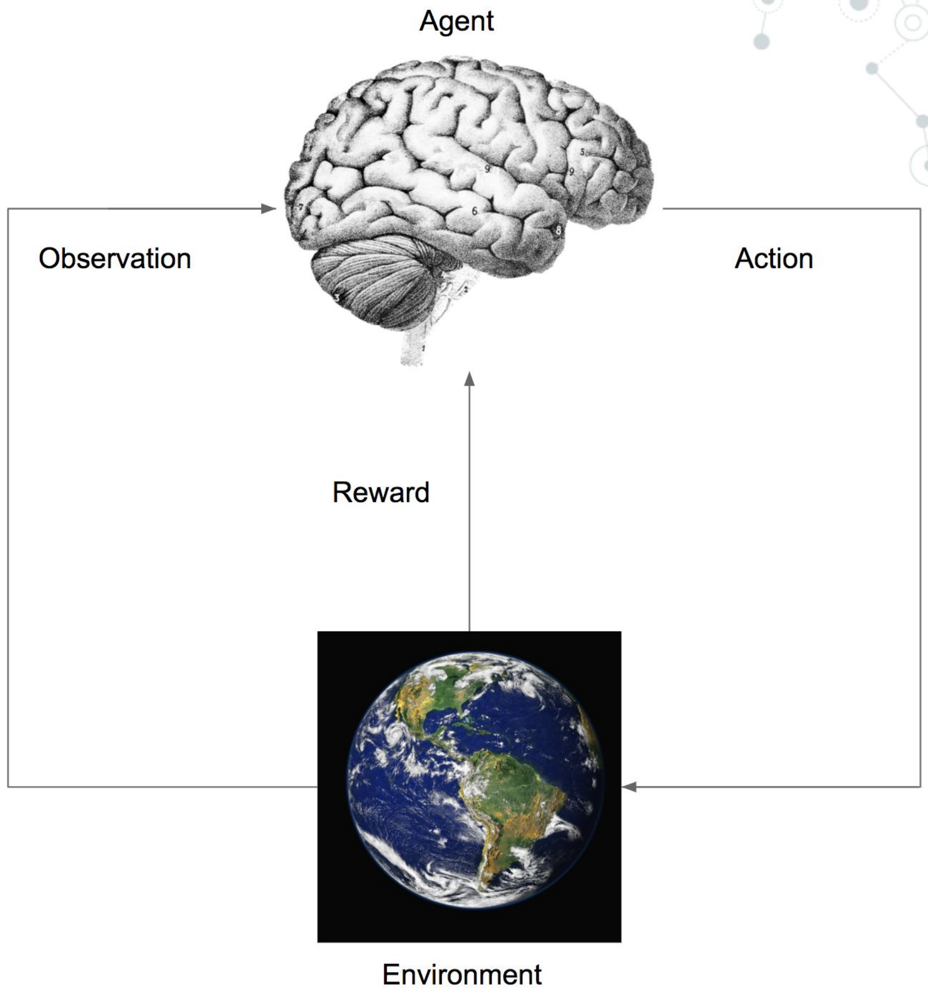


Problem Setup

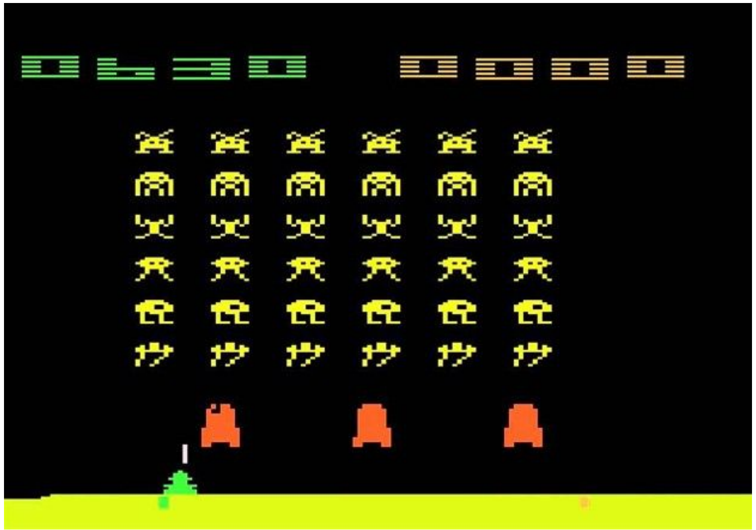
-Problem moves by time step

-At each step the agent sees a state and a reward.

-The agent uses that information to decide an action



Goal



The Value Function

-We want to maximize expected future reward.

-We also want to prefer immediate reward over delayed rewards.

$$V^p(s) = \sum_{k=1}^{+\infty} \gamma^{k-1} r_k = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

- ***γ*** is a hyperparameter between 0 and 1 and is called the “discount”.

-$\mathbf{r_t}$ is the reward at timestep **t**

-The Value Function maps a value to a state, but it does tell us what action to take.

Q-Values

-Solution: Estimate Q-Values instead! *Q(s,a)* returns the value of taking an action *a* in a state *s*.

-Now, it is easy to determine what action to take given a state.

$$\underset{a}{\operatorname{argmax}} \{Q(s,a)\}$$

Q-Learning

Initialise the Q' table with random values.

1. Choose an action a to perform in the current state, s.
2. Perform a and receive reward $\mathcal{R}(s, a)$.
3. Observe the new state, $\mathcal{S}(s, a)$.
4. Update:

$$Q'(s, a) \leftarrow \mathcal{R}(s, a) + \gamma \max_{\alpha} \{Q'(\mathcal{S}(s, a), \alpha)\}$$

5. If the next state is not terminal, go back to step 1.

-R(s,a) returns the reward of taking action *a* in state *s*

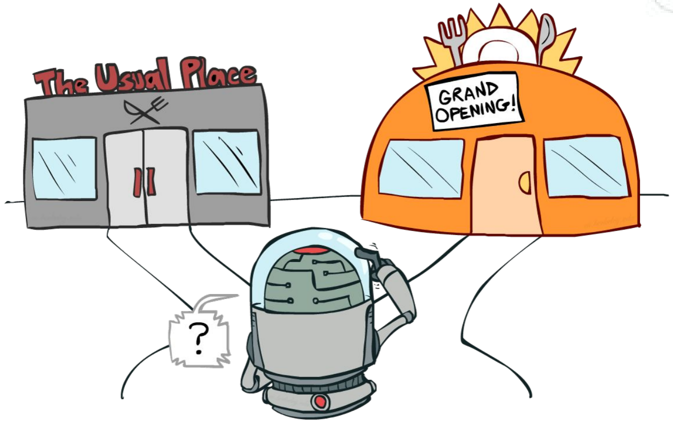
-S(s,a) returns the next state, *s'*, after taking action *a* in state *s*.

Exploration vs. Exploitation

-Exploring = less reward, but understand environment better

-Exploit = more reward, may be missing out on an important part of environment

- ε-greedy action selection: We take a random action with probability ε and decrease ε over time.



Shortcomings of Basic Q-Learning

-We do basic Q-Learning by storing the Q-Values in a table for each state-action pair.

-It does not work with continuous state spaces.

-It performs very poorly in very large state spaces.

-It requires a huge table as well for large state spaces.

-Tables do not scale well for larger problems in general.

-For example, for the DQN Atari setup we would need...

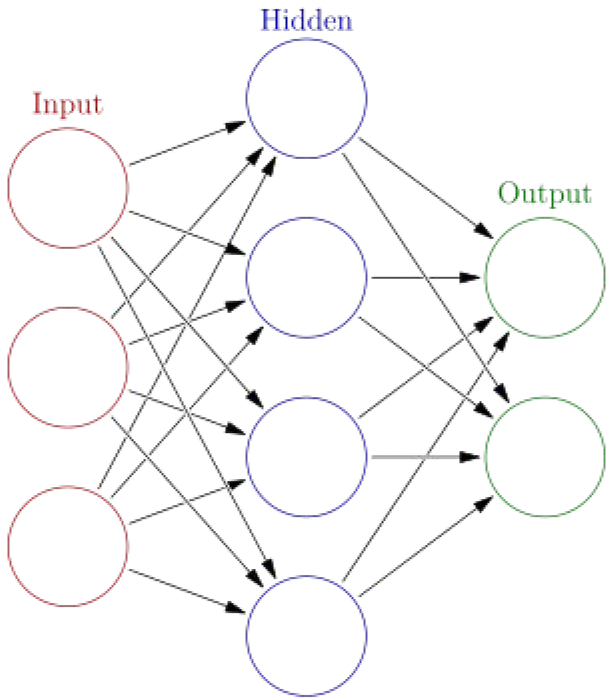
$$(84*84*4)*18 = 508032 \text{ Values}$$

Neural Networks!

On Neural Networks

-Neural Networks approximate a function given a sufficient number of inputs and outputs.

-We are trying to approximate the function Q(s,a).



Experience Replay

-The big innovation!

-We to store the SARS in a buffer and then randomly sample to train the network.

-Similar to how animals/humans recall previous experiences while learning. (https://www.nature.com/articles/nature14236)

Coding It All Up

Policy-Based Methods

-We just went over a Value-Based Method

-Learn Policy (function mapping states to action) directly instead of Q-Values in the neural network.

DDPG:

https://arxiv.org/abs/1509.02971

A3C:

https://arxiv.org/abs/1602.01783

PPO:

https://arxiv.org/abs/1707.06347

Where to go from here?

-There are some very simple and effective improvements on the code that we just wrote.

-Here are some easy papers you can read and implement with the code we just wrote!

Papers to Read

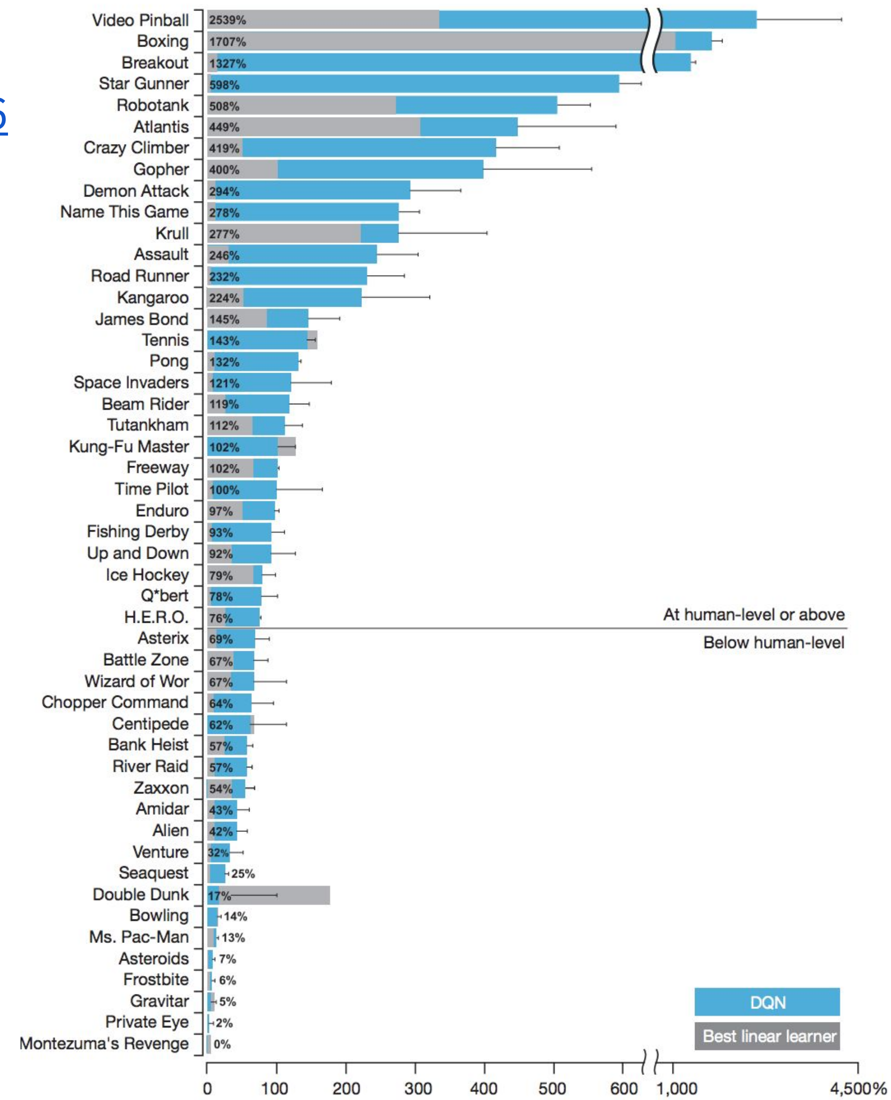
Human-level control through deep reinforcement learning

https://www.nature.com/articles/nature14236

-Nature, 2014 - rise of Deep RL

-Introduces DQN.

-Adds a “target network” to our implementation



Deep Reinforcement Learning with Double Q-learning

https://arxiv.org/abs/1509.06461

-September 2015

-Target Network to estimate Q-values, use online network for actions

-Prevent overstimation of state values

Dueling Network Architectures for Deep Reinforcement Learning

https://arxiv.org/abs/1511.06581

-November, 2015

-Estimate average value of a state, and change the value with each action.

-Finds proper action to take in a situation where the state is good

Parameter Space Noise for Exploration

https://arxiv.org/abs/1706.01905

-June, 2017

-Inject noise into parameters /weights of network.

-Better than \eps-greedy exploration

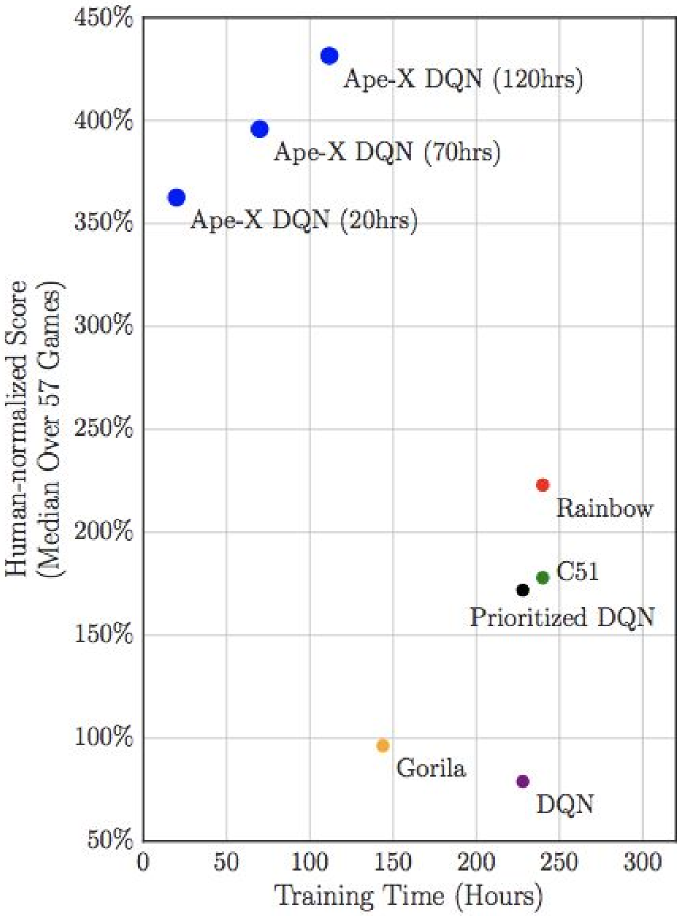
Distributed Prioritized Experience Replay

https://openreview.net/pdf?id=H1Dy---0Z

-Very new paper

-Large number of threads running the game at the same time to update the experience buffer.

-Uses “prioritized experience replay” which samples from the experience buffer based on how much it “learns” from the sample.



Active Areas

-Hierarchical Reinforcement Learning

Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation.

https://arxiv.org/abs/1604.06057

-Model-Based Reinforcement Learning

Learning model-based planning from scratch

https://arxiv.org/abs/1707.06170

-Improved Exploration

Curiosity-driven Exploration by Self-supervised Prediction

https://arxiv.org/abs/1705.05363

-Benchmarks and Environments

StarCraft II: A New Challenge for Reinforcement Learning

https://arxiv.org/pdf/1708.04782.pdf

Other Active Areas

-Multi-Agent RL

Multi-agent Reinforcement Learning in Sequential Social Dilemmas

https://storage.googleapis.com/deepmind-media/papers/multi-agent-rl-in-ssd.pdf

-Memory and Attention

Control of Memory, Active Perception, and Action in Minecraft

https://arxiv.org/abs/1605.09128

-Transfer Learning / K-Shot Learning

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

https://arxiv.org/abs/1703.03400

-Competitive Self-Play

Emergent Complexity via Multi-Agent Competition

https://arxiv.org/abs/1710.03748

-And a lot more!

Take-Aways

-Deep RL isn’t that complicated!

-There are tons of papers!